Unsupervised intrinsic calibration of depth sensors via SLAM

Alex Teichman Stephen Miller Sebastian Thrun

***Abstract*—We present a new, generic approach to the calibration of depth sensor intrinsics that requires only the ability to run SLAM. In particular, no specialized hardware, calibration target, or hand measurement is required. Essential to this approach is the idea that certain intrinsic parameters, identified here as *myopic*, govern distortions that increase with range.**

We demonstrate these ideas on the calibration of the popular Kinect and Xtion Pro Live RGBD sensors, which typically exhibit significant depth distortion at ranges greater than three meters. Making use of the myopic property, we show how to efficiently learn a discrete grid of 32,000 depth multipliers that resolve this distortion. Compared to the most similar unsupervised calibration work in the literature, this is a 100-fold increase in the maximum number of calibration parameters previously learned.

Compared to the supervised calibration approach, the work of this paper means the difference between A) printing a poster of a checkerboard, mounting it to a rigid plane, and recording data of it from many different angles and ranges - a process that often requires two people or repeated use of a special easel - versus B) recording a few minutes of data from unmodified, natural environments. This is advantageous both for individuals who wish to calibrate their own sensors as well as for a robot that needs to calibrate automatically while in the field.

I. Introduction

This work was originally motivated by a practical need. During an attempt to automatically discover the transform between two rigidly-mounted RGBD sensors, we found that, though the extrinsics were correct, the actual overlay of the two pointclouds contained significant disagreements. While the depth data from the Kinect or Xtion Pro Live sensors tends to be accurate out to about three meters, systematic warping of the depth data substantially degrades accuracy with increasing distance out to the maximum range of ten meters. The exact origin of this distortion is unknown due to the closed nature of the system design, and, as we will show, varies from one sensor to the next. Resolving this issue could improve the state of the art in any area that makes use of these sensors.

One might wonder if this is actually a valuable problem to solve. Might PrimeSense just release a new version of their software or hardware that dramatically improves the calibration? Due to the closed nature of their system it is hard to know if this distortion is intrinsically difficult to resolve or if they have only optimized for the close-range use case that they primarily care about. While it appears their focus is currently miniaturization rather than accuracy improvement [1], the answer is: Yes, this might happen. However, we can pose a more general depth sensor calibration algorithm *template*, of which PrimeSense distortion calibration is just one example. In this way, we can make a lasting contribution while simultaneously solving the community's more immediate need for a depth distortion solution for the popular PrimeSense sensors.

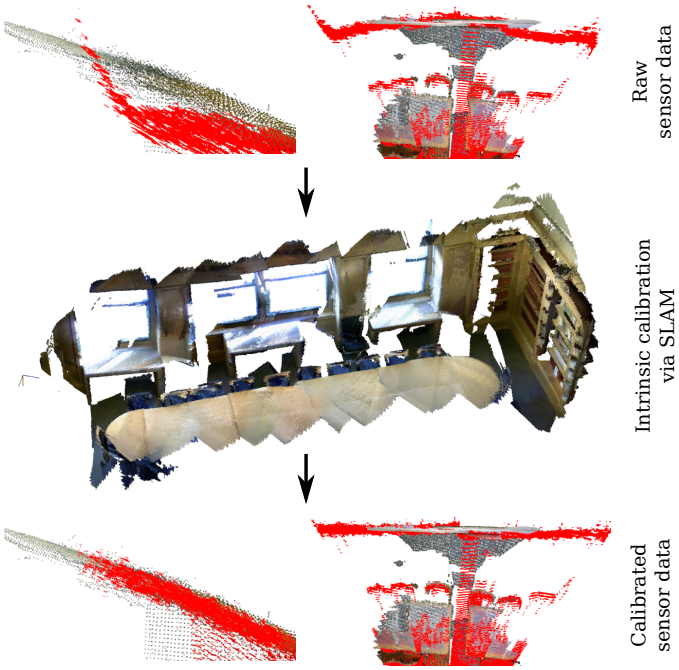

Fig. 1. Overview of our method calibrating an Asus Xtion Pro Live. Single frames of sensor data (red) are shown overlaid on an accurate reference map. Corner warping at close range and gross depth errors at long range are both corrected. Results for the Kinect are analogous.

The intuition behind our approach is simple. Using SLAM, one can build a map with relatively accurate close range data, then compare inaccurate long range data with the expected measurements from the map. For example, say the corners of a depth camera report ranges that are consistently 10% too far and we run SLAM looking at a wall from different distances. If the map is built with a maximum range of 2m, then the wall in the map will be wrong by at most 0.2m, and at 10m the observed discrepancy between the map and the measurement is 0.8m, close to the actual error. This information can be used to step the intrinsics towards their optimal values.

In summary, the primary contributions of this paper are that we

- propose a general, unsupervised intrinsic calibration approach (CLAMS: Calibrating, Localizing, and Mapping, Simultaneously) appropriate for many different depth

sensors and intrinsic calibration parameters.
- identify the *myopic* property of certain depth sensor intrinsics and show how to use it to fit calibration models with very large numbers of parameters.
- demonstrate these ideas on the calibration of the popular Kinect and Xtion Pro Live sensors using only a few minutes of recordings from unknown, unmodified environments.

Related work is deferred to the end so that comparisons with our method can be drawn.

II. The myopic property

A depth sensor intrinsic parameter is termed *myopic* if an erroneous setting of that parameter generates absolute error that increases with distance from the sensor. Many depth sensor intrinsic parameters have this property.

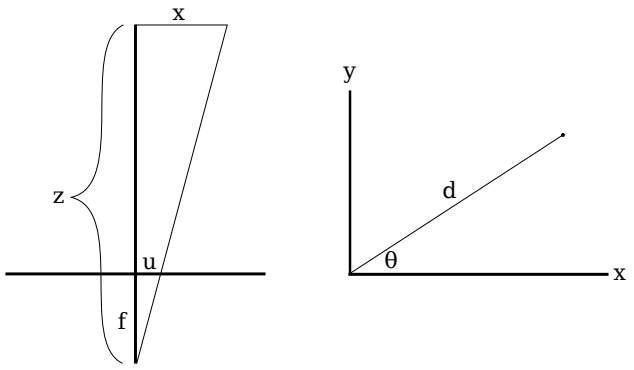

Fig. 2. Standard camera model (left) and laser range finder beam angle (right). Absolute error generated by incorrect focal length or incorrect beam angle grows with range in both these models; these are *myopic* intrinsic parameters.

Consider, for example, the standard pinhole camera model used with depth cameras, shown in Figure 2. If an incorrect focal length f' is used instead of f, the resulting depth point appears at (x', y', z') instead of (x, y, z). Because $z > 0$, the absolute error can be written as

$$\begin{aligned} e &= \sqrt{(x - x')^2 + (y - y')^2} \\ &= z\sqrt{(u/f - u/f')^2 + (v/f - v/f')^2}. \end{aligned}$$

So, absolute error increases linearly with distance from the sensor; focal length is *linearly myopic*. There is an analogous linear relationship for laser range finder beam angles. Assuming for simplicity that the beam lives in two dimensions, the absolute error resulting from using an incorrect angle θ' instead of θ is

$$e = d\sqrt{(\cos\theta - \cos\theta')^2 + (\sin\theta - \sin\theta')^2},$$

where $d > 0$ is the reported distance.

An example of a non-myopic intrinsic parameter is a constant depth offset term such as that used in Velodyne beam calibration [14], resulting from fixed timing offsets in the time-of-flight circuitry.

III. The generic CLAMS approach

The most generic version of CLAMS, hinted at in [14], is the optimization

$$\underset{T,C}{\text{minimize}} \quad f(T, C)$$

over a trajectory T and a calibration C, with a function f that measures some aspect of map error. Ideally, f would be convex in both parameters simultaneously, but unfortunately this case seems rare. Instead, we typically must settle for an approximate alternating optimization, where the objective function used for SLAM is different than the one used for calibration; convergence is not guaranteed. This approach relies on the reasonable assumption that improvements in calibration will result in improved SLAM solutions and vice versa.

Previous works (in which T was assumed to be given) have suggested solving for the calibration using various measures of pointcloud co-locality [18, 14]. The myopic property suggests a variation. Iteratively,

1) holding the calibration fixed, solve for the trajectory using SLAM.
2) construct a pointcloud - the *map* - from the trajectory using only relatively accurate short range depth data.
3) optimize the calibration using expected measurements generated from the map.

The specific form of step 3 could vary depending on what parameters are being calibrated. For example, optimization of the focal length of a depth camera or beam angle of a laser could be simple grid search where the objective function measures the average depth discrepancy between the measured depth images and the expected depth images generated from the map. In other cases, such as estimating a per-pixel depth multiplier, maximum likelihood estimation of the intrinsic parameters can be done in closed form. It is this latter case that we will explore in the following sections.

IV. Depth distortion in PrimeSense RGBD sensors

Here we focus on the Asus Xtion Pro Live and Microsoft Kinect, referred to collectively as "PrimeSense sensors" as they are both derived from the same design by PrimeSense.

Fortunately, the absolute depth error in question increases with range, which indicates there is an unmodeled myopic intrinsic parameter. This can be seen by observing depth data of a flat wall as seen from above at different distances. At close range, the data is highly planar, and at longer ranges significant warping is evident. It can further be verified with a tape measure that the absolute position of the plane tends to be accurate at close range. See Figure 3 for a characteristic example of this distortion.

To model the effects of this distortion, one could try using a *depth multiplier image*, where each element of a raw depth image must be multiplied by a value that depends on the image coordinates. Like focal length and beam angle, the elements of a depth multiplier image are linearly myopic parameters. Consider a single depth pixel which reports a value

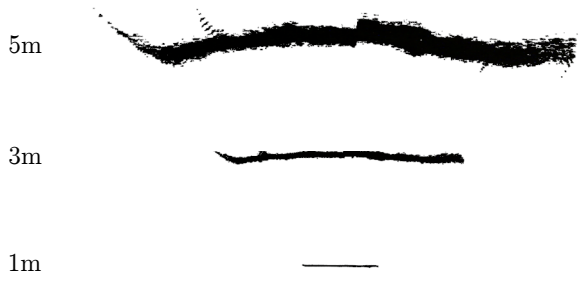

Fig. 3. Raw PrimeSense data of a flat wall, displayed to scale, seen from above. It is this type of distortion that we calibrate out in Section IV.

of $\tilde{z} > 0$ and the one element m of the depth multiplier image that applies to it. Using an incorrect value m' instead of m generates a point $(x', y', m'\tilde{z})$ rather than the correct point $(x, y, m\tilde{z})$; the resulting error is

$$\begin{aligned} e &= \sqrt{(x-x')^2 + (y-y')^2 + (m\tilde{z} - m'\tilde{z})^2} \\ &= \tilde{z}\sqrt{\left(\frac{um}{f} - \frac{um'}{f}\right)^2 + \left(\frac{vm}{f} - \frac{vm'}{f}\right)^2 + (m-m')^2}. \end{aligned}$$

Empirically, however, the distortion appears to be super-linear, so we use several depth multiplier images in different range brackets. Formally, the distortion model that we will fit is

$$z = \sum_i \mathbf{1}\{\tilde{z} \in [z_i, z_{i+1})\} D_i(u, v)\tilde{z},$$

where $\tilde{z}$ is the raw measurement reported by the sensor, the D_i are the individual depth multiplier images, and $z_0, z_1, ..., z_5$ are $0, 2, ..., 10$. To avoid discrete jumps in the calibrated data at runtime, we apply linear interpolation along the beam. Finally, we discretize the depth multiplier images into bins of 8×6 pixels. At 640×480 resolution and a maximum range of 10m, this results in 32,000 parameters.

A. Algorithm

Making use of the myopic property, the outline of the algorithm is to

1) collect several minutes of raw sensor data containing sufficient coverage in all regions of the view frustum.
2) find the sensor *trajectory* with SLAM.
3) construct a pointcloud (the *map*) using only sensor data that was seen from less than two meters away.
4) harvest training examples from the map and fit the depth multiplier images in closed form using maximum likelihood estimation.

It is probably beneficial to iterate on steps 2-4, as improvements to the calibration model should produce improvements in the SLAM trajectory. For this particular problem, however, it is possible to produce a reasonable solution in a single iteration, probably because PrimeSense data is accurate at close range and our SLAM solution works sufficiently well with the raw sensor data. Details on each of the steps follow.

Data collection - Calibration requires training data in all parts of the view frustum that one cares about. In our case, we wish to calibrate the entire view frustum, so we walk back and forth along an approximately ten meter path and wave the sensor around at random so that the edges and corners will get sufficient data even at long range. Walking back and forth over the same path allows the SLAM algorithm to find loop closures that improve the final accuracy of the map. Typically, several sequences like this are collected in different locations, resulting in a few minutes of training data. In our experiments, we use natural, unmodified office environments.

SLAM - Our SLAM approach is similar to that of RGBD-SLAM [6, 5] or RGB-D Mapping [9, 8], but with some modifications. Edges in the graph are determined with a grid search that considers depth, color, and smoothed image gradients, with the relative weighting chosen by cross validation. After the SLAM solver runs, we remove edges that have high error and edges that have no error. The latter are edges that are unconfirmed by loop closures. We take the largest connected subgraph that results. The general graph optimization package g2o [13] is used as the core SLAM solver.

Map building - Once a trajectory is known, we build a pointcloud referred to as the *map* using only close range data. Specifically, for each pose in the trajectory, we project into the map each point that was observed from less than two meters. In this way, we can be reasonably confident in the accuracy of the map. To keep the number of points manageable, we use a voxel grid filter with 1cm resolution. See Figure 4 for example maps used for calibration.

Model fitting - Each parameter is fit independently assuming the measured depth $\tilde{z}$ is generated by applying a multiplier w to the ground truth depth z, then adding Gaussian noise. Formally, we have

$$\mathbb{P}(\tilde{z}|z, w) = \eta \exp\left(\frac{-(\tilde{z} - wz)^2}{2\sigma}\right).$$

This results in a maximum likelihood estimate of

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & \prod_{i=1}^{M} \mathbb{P}(\tilde{z}_i|z_i, w) \\ \underset{w}{\text{minimize}} \quad & \sum_{i=1}^{M} (\tilde{z}_i - wz_i)^2 \\ w &= \frac{\sum_i z_i \tilde{z}_i}{\sum_i z_i^2}. \end{aligned}$$

As some parameters may not get any training examples, we assume the existence of a single example with $z_i = \tilde{z}_i = 1$. At runtime, we are interested in predicting ground truth depth from measured depth, so we populate the appropriate bin of the appropriate depth multiplier image with $1/w$.

To collect a training set, we iterate over all frames in the trajectory and all points in the frame, recording measured range and map range. For a given frame and pixel location, $\tilde{z}_i$ is just the range that was reported by the sensor. The map

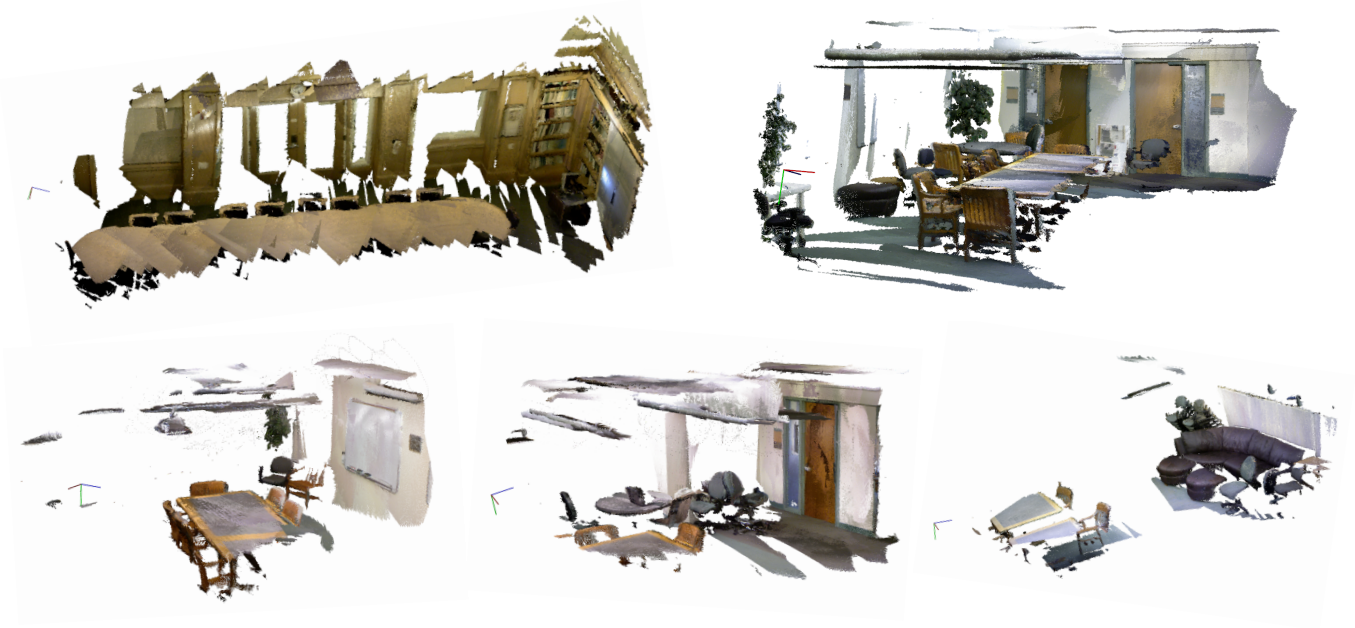

Fig. 4. Example maps used for calibration.

range z_i is determined by averaging the range of nearby points in the map. Specifically, we define a cone that contains the 3D point and a 2cm radius around it in the (x, y) plane, i.e. parallel to the camera plane. The mean range of points in the cone with less than 20% deviation from the measured range $\tilde{z}_i$ is used for z_i. The training example is rejected if the standard deviation of these points is greater than 3cm or if the map points are near the edge of the map frame.

This filtering helps attenuate undesirable effects of occlusion and slight angular errors in the trajectory. Moreover, if the view of a surface is oblique and the map contains minor SLAM errors, the filtering will tend to set the expected range z_i such that these errors average out. Points at the edge of the map are rejected because the mean will tend to be inaccurate if the view is oblique.

V. PrimeSense calibration evaluations

We present two different quantitative evaluations along with qualitative results demonstrating that the CLAMS algorithm of Section IV resolves the depth distortion in PrimeSense sensors. All experiments use the RGB-to-depth mapping provided by the factory. Quantitative experiments focus on the Xtion Pro Live sensors `xpl00` and `xpl01`, with qualitative results shown for several more sensors, including three Kinects, in Section V-C. SLAM maps were checked by eye to confirm no obvious trajectory errors existed. Unfortunately, even using state of the art techniques, mapping failures do still sometimes occur.

A. Flat wall evaluation

Our first evaluation is fairly simple: When the sensor sees only a flat wall, how flat is its output?

We calibrated `xpl00` using about 4.5 minutes of raw data as described in Section IV-A. Example maps used for calibration are shown in Figure 4. The test sequence was collected by walking a handheld sensor towards a large, flat wall, starting from a distance of about five meters. The field of view of the sensor included only the wall, so we know that ideal sensor output should have all points in each view lying on a plane.

To evaluate planarity, we fit a normal vector n and offset b using a RANSAC method followed by refinement using SVD on the inliers. RMS error, $\sqrt{\frac{1}{M}\sum_i (n^T p_i - b)^2}$, was recorded for all points for every 15th frame in the sequence, with and without the depth multiplier images learned via CLAMS. Results are shown in Figure 5. The sensor output is substantially more planar using the learned calibration.

Even with calibration, RMS error increases somewhat with range. In Figure 5, calibrated data at 5m is highly planar, but forms a thicker line than at close range. Depth discretization inherent in the raw sensor data, visualized in Figure 6, generates most of this error.

The learned depth multiplier image for the 4-6m slice is shown in Figure 7. It is possible to see the correspondence between the model and the distortion in the uncalibrated data. In particular, the top left and bottom left corners of the multiplier image show that the raw data needs to be pulled closer to the sensor, corresponding to two most noticeable "peeling" effects at five meter range in Figure 6.

This experiment is easy to understand, visualize, and quantitatively evaluate, but can only evaluate out to five meters due to the requirement of having a very large plane available. Further, if one considers this experiment in isolation, it is unlikely but possible that while the calibrated output is planar,

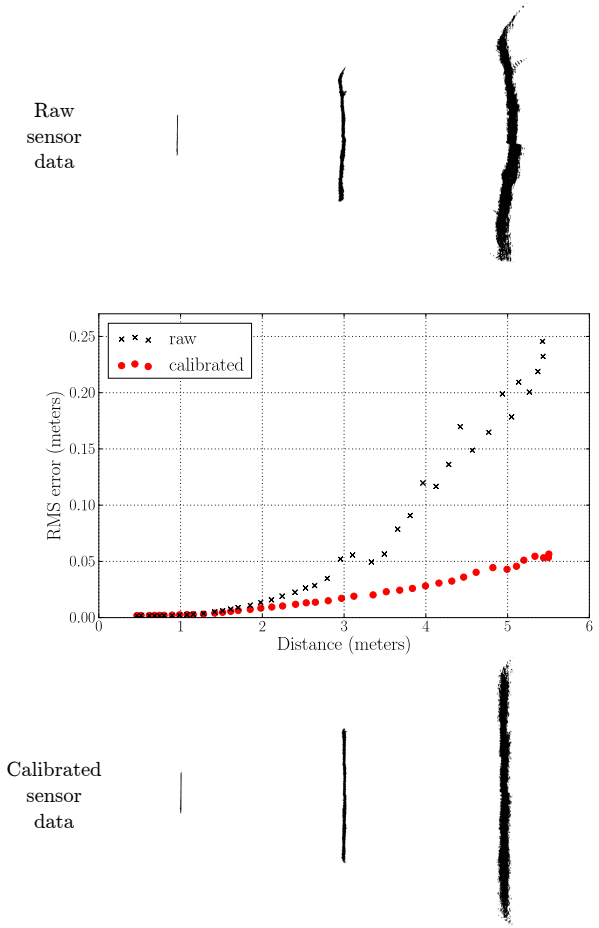

Fig. 5. Flat wall evaluation. Quantitatively, deviation from planarity is substantially reduced using the intrinsics learned with CLAMS. Qualitative results show sensor data of the flat wall as seen from above. Applying the learned calibration produces planar output; the increase in thickness and RMS error at long range in the calibrated data can be attributed to the depth discretization inherent in the sensor, visualized in Figure 6.

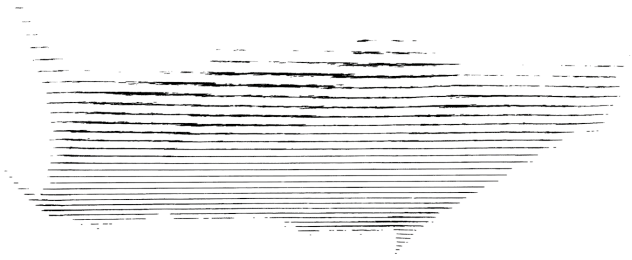

Fig. 6. Alternate view of the raw sensor data shown in the upper right of Figure 5, showing the same flat wall from approximately above. The left side of the image is the left side of the field of view of the sensor. Depth discretization is obvious at this angle. This discretization results in "fuzzier" depth data at longer ranges, even when the calibration is correct.

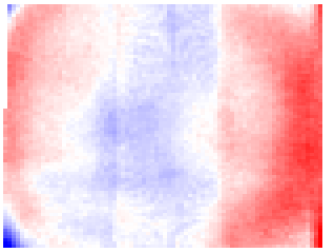

Fig. 7. Learned depth multiplier image for sensor `xpl00`. White indicates a depth multiplier of one, full blue of 0.9, and full red of 1.1. The top left and bottom left corners of the multiplier image show that the raw data needs to be pulled closer to the sensor, corresponding to the two most noticeable "peeling" effects in raw sensor data at five meters in Figure 6.

the actual plane is incorrect. These drawbacks are eliminated in the following evaluation.

B. Unseen test maps evaluation

In this evaluation, we make use of SLAM to generate test data. This gives us confirmation independent of Section V-A that the calibration procedure is working. Additionally, we show that the required amount of calibration data is 2-3 minutes and that the results are consistent across varying calibration datasets. Both `xpl00` and `xpl01` are evaluated in this section.

Specifically, we constructed several maps for each sensor[1] using steps 2 and 3 from Section IV-A and assigned them randomly to calibration and testing sets. Because only close range data is used and maps are checked for obvious errors, we can have reasonable confidence in the accuracy of these maps. At calibration time, random subsets of the calibration maps were selected and used for calibration, followed by an evaluation on all test maps. For each observed depth pixel, we computed the best distance estimate from the map using the filtering of Section IV-A. As in Section V-A, RMS error was measured for both raw and uncalibrated sensor data.

Error measurement in this experiment requires some consideration. In Section V-A, despite qualitative results that show that the calibration restored the planarity of the wall, RMS error still increased with range due to depth discretization. Here, we have an analogous but somewhat worse situation: Total depth error includes SLAM trajectory error, occlusion misprediction, depth discretization, and depth distortion. As a result, even the optimal intrinsics cannot produce 100% error reduction.

What we *can* show from the quantitative results is that A) error reduction is significant and stable over different sets of calibration maps, and B) one needs 2-3 minutes of calibration data to expect to mostly saturate performance, though increasing amounts of data are still beneficial. These quantitative results are shown for both sensors in Figure 10.

Qualitative results for both sensors are shown in Figure 8. Detailed discussion is provided in the caption.

[1]Nine for `xpl00`; seven for `xpl01`.

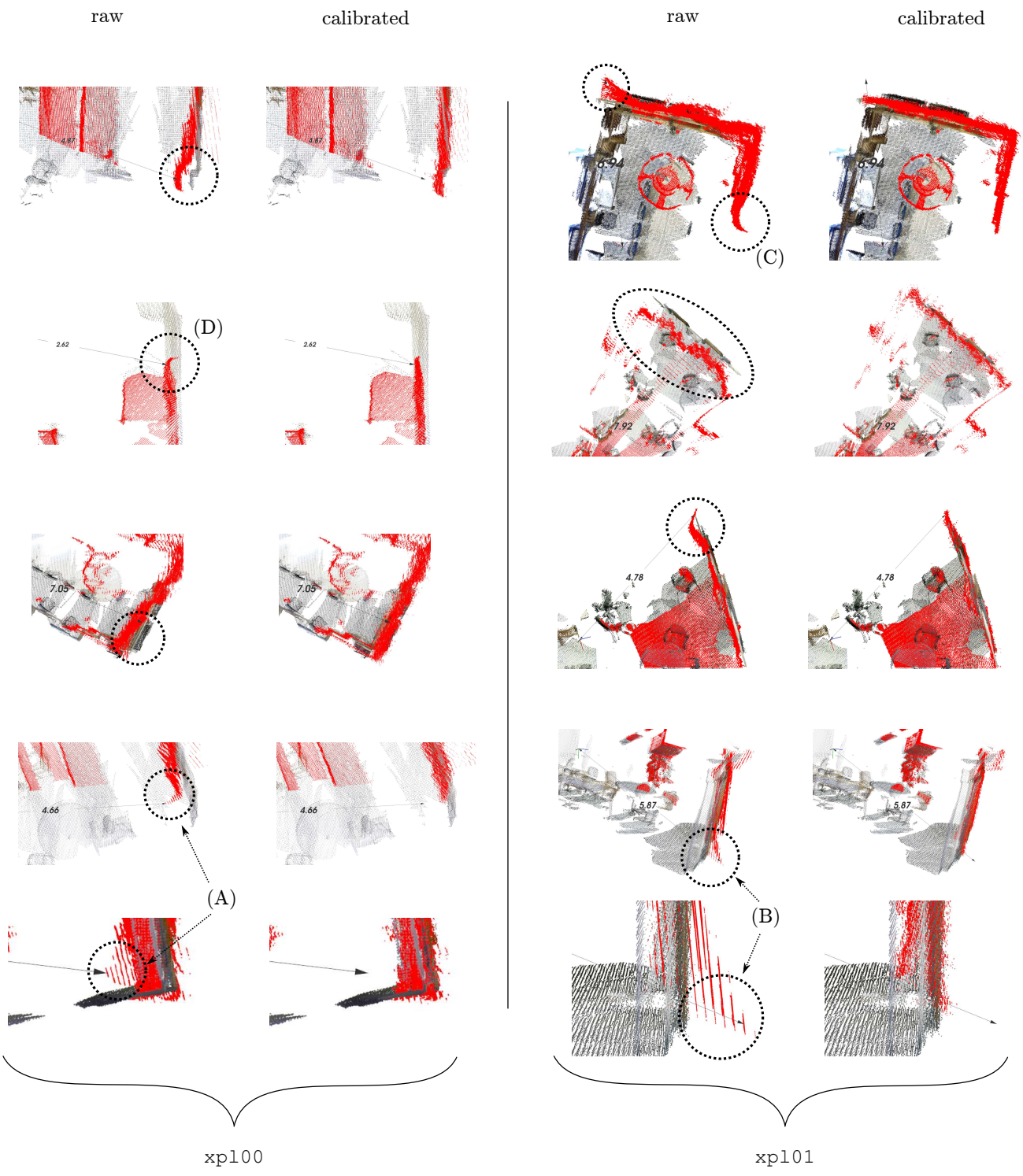

Fig. 8. Qualitative results showing the effects of calibration overlaid on reference maps. Best viewed on-screen and zoomed in. These visualizations show an unseen test map with depth data from a single frame overlaid in red; an ideal sensor would produce red points that match up with the map points. Numeric annotation of arrows in the pointclouds are of Euclidean distance to the sensor in meters. (A) Two different perspectives of the same scene show that the bottom right corner of raw sensor data from `xpl00` is too close to the sensor and must be pushed further away to produce a flat wall. (B) Similarly, here we have two different perspectives of the same scene for `xpl01`. In this sensor, the bottom right corner has the opposite distortion - it must be pulled in to produce a flat wall. This can be confirmed in the depth multiplier images in Figure 9. (C) In this scene, `xpl01` was held upside down, so the two circled regions correspond to the bottom corners of the depth multiplier image in Figure 9. (D) Even at close range, corner warping can be observed in the raw data and suppressed with CLAMS.

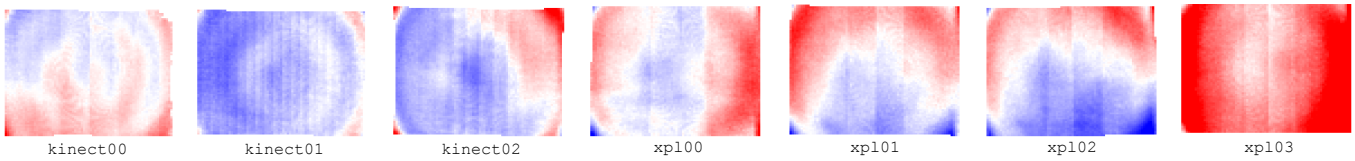

Fig. 9. Depth multiplier images for several sensors following the same coloring scheme as in Figure 7. The significant variation shown here validates the need for independent sensor calibration. `xpl03` is a newer version of the Xtion Pro Live sensor, identifiable by its square rather than oval projector aperture. `xpl01` and `xpl02` are perhaps from the same batch.

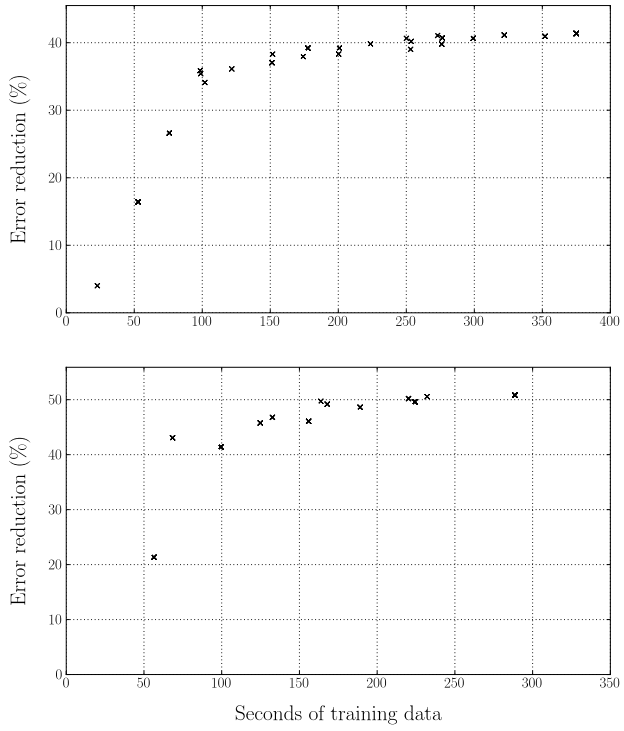

Fig. 10. Depth error reduction evaluated on unseen test maps. Depth error includes SLAM trajectory error, occlusion misprediction, depth discretization, and depth distortion; thus, 100% error reduction is not possible even if the exact depth distortion model were available. We can conclude from this plot that A) error reduction is significant and stable over different sets of calibration maps, and B) one needs 2-3 minutes of training data before accuracy improvement becomes slow. Sensors `xpl00` (top) and `xpl01` (bottom) are shown.

C. Sensor variation

To validate the need for individual sensor calibration, here we show that all PrimeSense sensors do not share the same set of intrinsic parameters. We calibrated several different sensors, confirmed their error reduction and qualitative results were reasonable, and plotted the 4-6m depth multiplier image for each in Figure 9. Only `xpl01` and `xpl02` seem similar enough that they might share the same model; they are perhaps from the same manufacturing batch. The works of Smisek et al. [19] and Herrera C. et al. [10] discuss vertical banding in the depth distortion, which we recover here in all the sensors we tested.

D. Timing results

Runtime of the calibration procedure is asymptotically linear in the amount of calibration data. At a high level, the algorithm is just SLAM followed by an update of the distortion model, run for S different sequences. Crucially, the length of each of these sequences is bounded by a constant; each sequence need only include data out to the maximum range of ten meters. For a single sequence, the runtime of both SLAM and model updating depends only on the length of the sequence in question, and therefore is also bounded by a constant. Thus, the calibration procedure as a whole is just $O(S)$; if you decide you need a more accurate calibration, you can simply collect more bounded-length sequences and expect that runtime will grow only linearly.

Given the sensor trajectory, the average calibration time in Section V-B was ten minutes on a Xeon X5650. However, our implementation of SLAM is relatively slow and full sensor calibration including SLAM for several sequences is generally an overnight procedure. We expect that this can be substantially improved, both with code optimizations for the handheld calibration scenario and by making use of additional problem data for other scenarios.

Applying the calibration model to an individual 640×480 depth image takes about 2ms on a modern laptop with an i7-3820QM processor, making this distortion model practical to use online.

VI. Related work

We are unaware of any previous work on depth sensor intrinsic calibration that requires only handheld recordings of depth sensor data in unmodified environments. Broadly speaking, related work either makes use of specialized calibration targets or applies to different calibration problems.

Depth distortion calibration methods using calibration targets cover RGBD sensors such as the Kinect or Xtion Pro Live and time-of-flight sensors such as the PMDTechnologies device. Herrera C. et al. [10] use a checkerboard attached to a plane to acquire calibration data; this is probably the most similar work in the supervised category, as they make use of a similar depth distortion model. Zhang and Zhang [20] largely discuss the 6DOF transform between the RGB and IR cameras on a Kinect, but also consider a simple depth distortion model with two parameters. Time-of-flight (ToF) sensor intrinsic calibration using calibration targets has been more well-studied on its own [15, 16, 7] and in the context of

sensor fusion [21, 11]. ToF distortion models typically take a somewhat different form, where the depth distortion is taken to be a function of only distance, rather than distance and image location as in this work and Herrera C. et al. [10].

Unsupervised calibration methods exist for several related, but different problems. Most similar in this category is probably the work of Levinson and Thrun [14], who demonstrate automatic calibration of the beam angles, distance-response functions, and sensor-to-vehicle extrinsics of a Velodyne laser range finder mounted on an autonomous vehicle. However, they assume the relative vehicle trajectory is provided by a high accuracy GPS / IMU / wheel encoder system; this is a reasonable assumption for calibration of autonomous driving systems, but does not hold for hand-held sensors or most robots. Similarly, Sheehan et al. [18] demonstrate unsupervised calibration of a custom Velodyne-like sensor, making use of the known spinning motion of the sensor base to automatically calibrate beam angles, relative laser positions, and timing offsets. In a similar vein, there are several unsupervised methods that demonstrate extrinsic calibration between multiple depth sensors by making use of sensor motion [2, 17]. In the realm of RGB-only cameras, Carrera et al. [3] show extrinsic calibration between several cameras on a moving platform, and Civera et al. [4] compute camera intrinsics using structure from motion. Finally, in a more robotics-specific application, Kümmerle et al. [12] solve for sensor mount position and kinematic parameters such as wheel radius during SLAM.

Among unsupervised methods for different calibration tasks, the largest number of parameters calibrated of which we are aware is 192, where two beam angles and one range offset is computed for each of 64 Velodyne beams [14]; other works in this category tend to calibrate on the order of 10 parameters. The objective function used in [14] is essentially minimization of point to plane distance and encodes a similar intuition as the "crispness" objective function of [18]. Optimization is grid search where each change to a parameter requires updating nearest-neighbor lookup structures and surface normals. Using this same approach with our 32,000 parameters rather than 192 is probably infeasible. Similarly, it is not clear that optimizing the non-convex "crispness" objective function with so many more parameters would be tractable. It is the use of the myopic property and a simple generative model that allow us to find a good fit for two orders of magnitude more parameters in a reasonable amount of time.

VII. Conclusion

We have presented a generic approach to depth sensor calibration which requires no special calibration targets or measurements of the environment, just the ability to run SLAM. Further, we have shown how this approach can be applied to resolve the depth distortion inherent in the Xtion Pro Live and Kinect sensors. From a practical perspective, this result makes RGBD calibration require that a user only record a few minutes of data from natural environments - a substantial improvement over what used to be an onerous process.

Acknowledgments

Thanks to Jesse Levinson for the name "CLAMS" and to Jake Lussier for the voiceover of our video submission.

References

[1] PrimeSense unveils Capri, worlds smallest 3D sensing device at CES 2013, December 2011. URL http://www.primesense.com/news/primesense-unveils-capri/.

[2] Jonathan Brookshire and Seth Teller. Extrinsic calibration from per-sensor egomotion. In *RSS*, 2012.

[3] Gerardo Carrera, Adrien Angeli, and Andrew J. Davison. SLAM-based automatic extrinsic calibration of a multi-camera rig. In *ICRA*, 2011.

[4] Javier Civera, Diana R. Bueno, Andrew J. Davison, and J. M. M. Montiel. Camera self-calibration for sequential bayesian structure from motion. In *ICRA*, 2009.

[5] Felix Endres, Jurgen Hess, Nikolas Engelhard, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, 2012.

[6] Nikolas Engelhard, Felix Endres, Jurgen Hess, Jurgen Sturm, and Wolfram Burgard. Real-time 3D visual SLAM with a hand-held RGB-D camera. In *RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, 2011.

[7] Stefan Fuchs and Gerd Hirzinger. Extrinsic and depth calibration of ToF-cameras. In *CVPR*, 2008.

[8] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *ISER*, 2010.

[9] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. In *IJRR*, 2012.

[10] Daniel Herrera C., Juho Kannala, and Janne Heikkila. Joint depth and color camera calibration with distortion correction. In *PAMI*, 2012.

[11] Y.M. Kim, D. Chan, C. Theobalt, and S. Thrun. Design and calibration of a multi-view tof sensor fusion system. In *Computer Vision and Pattern Recognition Workshops*, 2008.

[12] Rainer Kümmerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous calibration, localization, and mapping. In *IROS*, 2011.

[13] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *ICRA*, 2011.

[14] Jesse Levinson and Sebastian Thrun. Unsupervised calibration for multi-beam lasers. In *ISER*, 2010.

[15] Derek Lichti and Changjae Kim. A comparison of three geometric self-calibration methods for range cameras. In *Remote Sensing*, 2011.

[16] Marvin Lindner, Ingo Schiller, Andreas Kolb, and Reinhard Koch. Time-of-flight sensor calibration for accurate range sensing. In *Computer Vision and Image Understanding*, 2010.

[17] Will Maddern, Alastair Harrison, and Paul Newman. Lost in translation (and rotation): Rapid extrinsic calibration for 2D and 3D LIDARs. In *ICRA*, 2012.

[18] Mark Sheehan, Alastair Harrison, and Paul Newman. Self-calibration for a 3D laser. In *IJRR*, 2012.

[19] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3D with kinect. In *IEEE Workshop on Consumer Depth Cameras for Computer Vision*, 2011.

[20] Cha Zhang and Zhengyou Zhang. Calibration between depth and color sensors for commodity depth cameras. In *International Workshop on Hot Topics in 3D*, 2011.

[21] Jiejie Zhu, Liang Wang, Ruigang Yang, and James Davis. Fusion of time-of-flight depth and stereo for high accuracy depth maps. In *CVPR*, 2008.